

## **REMARKS/ARGUMENTS**

This letter is in reply to the Office Action dated September 30, 2009.

### **Claim Amendments**

To expedite prosecution of the application, claims 1, 7 and 12 have been amended. **Claims 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17 and 19-24** remain in this application. Claims 1, 7 and 12 are independent claims.

To expedite prosecution of the application, and without prejudice, claim 1 has been amended to recite that the generated files are directly interpretable by a Java Virtual Machine. Support for this amendment can be found at paragraphs [0002] and [0016], for example, in the Applicants' description as filed. Applicants have made amendments to claims 7 and 12 that are analogous to the amendments made to claim 1.

### **Claim Rejections - 35 U.S.C. §103**

Claims 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17 and 19-24 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Swetland (U.S. Publication No. 2002/0170047 A1), in view of Baentsch et al. (WO 99/49392, "Baentsch"), further in view of Maatta et al. (NPL "Building AS/400 Client/Server Applications with Java", "Maatta"). The Applicants respectfully traverse all rejections.

### ***The Examiner has the burden to show all elements and an impetus to combine them***

To make a prima facie case under section 103, the Examiner's burden includes (but is not limited to) citing references that teach or suggest all of the features of a claimed invention: e.g. *In re Ochiai*, 71 F.3d 1565, 1572 (Fed. Cir. 1995); *In re Wada and Murphy*, Appeal No. 2007-3733, slip op. at 7 (BPAI 2008).

***None of the cited references disclose that the two or more files generated are directly interpretable by a Java Virtual Machine***

In the office action, the Examiner conceded that Swetland and Baentsch fail to disclose "[the computing unit configured to execute software] for generating two or more files from the plurality of class files". The Examiner indicated that Maatta was introduced to overcome this deficiency. However, the Applicants note that Maatta merely teaches how to archive and split Java applications using the .jar or .zip archive formats.

The Applicants explicitly identified deficiencies in the Java archive approach, for example, at paragraphs [0002] of the application as filed:

Java .class files may be archived (and optionally compressed) into a .jar file. However, .jar files are **not directly interpretable by the Java VM**, and the .class files must be extracted (and decompressed, if applicable) from the .jar file (and read into memory) in order for them to be linked, resolved and interpreted by the Java VM. (emphasis added)

Since .jar and .zip archives are not directly interpretable by a Java VM, and since Maatta only discloses splitting .jar or .zip archives, it is submitted that Maatta fails to disclose generating two or more files from the plurality of class files ... wherein the generated files are directly interpretable by the Java Virtual Machine.

***Maatta also fails to disclose sibling files, each comprising a sibling list for listing other sibling files in the common sibling group***

The Examiner also took the position that Maatta discloses "at least two of the generated files are generated as sibling files in a common sibling group, wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group". The Applicants respectfully traverse this position.

Maatta teaches that the `-split` function splits a source .jar or .zip archive into smaller .jar or .zip files. However, Maatta fails to disclose that **each** smaller .jar or .zip file comprises a list of other smaller files in the group, for example. Rather, Maatta explicitly discloses that "[n]o zip entries are added or excluded" and "[t]he entries in the source jar or zip file are **simply distributed** among the destination jar or zip files" (see pgs. 367 and 370, emphasis added). Accordingly, it is respectfully submitted that Maatta fails to disclose "at least two of the generated files are generated as sibling files in a common sibling group, wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group".

***The skilled person would not be motivated to combine the references cited by the Examiner in the proposed manner, because the proposed combination would be directed to a solution that teaches away from Applicants' claimed embodiments***

Although .jar files comprising archived and compressed .class files are smaller than the .class files themselves, storage space for the extracted (and decompressed, if applicable) .class files needs to be available in the environment where the application is to be executed, so that the Java VM may access the .class files. Consequently, a solution involving .jar files may not represent a savings in storage space. In fact, a solution involving .jar files may require extra storage space, for both the .jar files and the extracted .class files. The extracted .class files need not be retained once they have been loaded into memory, linked and resolved. However, both the .jar file and the in-memory representations of the .class files must be retained. In an environment having limited storage, where storage space is at a premium, it may therefore be preferable to store only the .class files and not to use a solution involving .jar files (see e.g. paragraph [0003] of Applicants' description).

Since the .class files must be extracted from .jar files prior to interpretation by the Java VM, the use of .jar files to split a single large .class file would not be as beneficial in certain situations, such as in implementations involving a flash memory whose physical

layout imposes an upper limit on file size that is exceeded by the .class (or .cod) file, for example (see e.g. paragraph [0029] of Applicants' description).

In contrast, the Applicants recognized that it would be advantageous, as an alternative to the exclusive use of solo .cod files in the representation of an application, to use sibling .cod files when the .cod files are to be stored in a storage medium that imposes a limit on the size of individual .cod files, for example. Since the .cod files retain the ability to be directly interpreted by the Java VM, a number of potential drawbacks of the .jar approach – too-large .class file sizes and additional required space for extraction – may be avoided.

Therefore, the Applicants submit that the proposed combination is directed to a solution that teaches away from the claimed subject matter, and accordingly, the skilled person would not be motivated to combine the cited references in the manner proposed by the Examiner.

## **Conclusion**

In view of the foregoing, the Applicants respectfully submit that amended claim 1 recites subject matter that is both novel and non-obvious over the cited references. Furthermore, since claims 7 and 13 recite analogous features, and the remaining claims are dependent, either directly or indirectly, on one of independent claims 1, 7 and 13, it is respectfully submitted that the subject matter of these claims is also novel and non-obvious for at least the same reasons. Withdrawal of the remaining rejections under 35 U.S.C. §103 is respectfully requested.

As previously noted, it is respectfully submitted that the cited documents do not teach all the features of the amended independent claims. Furthermore, they fail to provide a teaching, suggestion or motivation to combine features of the documents to arrive at the subject matter of the amended independent claims. The Applicants note that the Supreme Court's KSR decision<sup>1</sup> did not reject the use of a "teaching, suggestion or motivation" analysis as part of an obviousness analysis. The Supreme Court characterized the analysis as a helpful insight. It is respectfully submitted that the absence of a teaching, suggestion or motivation is a significant point in the Applicants' favor, as this absence is indicative of non-obviousness.

Although the Supreme Court did not reject use of a "teaching, suggestion or motivation" analysis, the Supreme Court did say that it was not the only possible analysis of an obviousness question. In the event that the Examiner chooses to pursue a different avenue for rejection, the Examiner is invited to explicitly identify the rationale and articulate the reasons on which such rejection is based, and it should be noted that any new avenue would be a new ground for rejection not due to any action by the Applicants.

The Applicants further respectfully remind the Examiner that, even after KSR, the following legal principles are still valid, having been endorsed by the Supreme Court or having been unaffected by its decision: (1) the USPTO still has the burden of proof on the issue of obviousness; (2) the USPTO must base its decision upon evidence, and it must support its decision with articulated reasoning; (3) merely demonstrating that all elements of the claimed invention exist in the prior art is not sufficient to support a determination of obviousness; (4) hindsight has no place in an obviousness analysis; and (5) Applicants are entitled to a careful, thorough, professional examination of the claims.

---

<sup>1</sup> *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398 (2007).

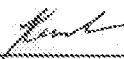
Appl. No. 10/536,745  
Amdt. dated December 22, 2009  
Reply to Office Action of September 30, 2009

In view of the foregoing comments, it is respectfully submitted that the pending claims in the subject application are now in condition for allowance, and a notice to that effect is respectfully requested.

Respectfully submitted,

BERESKIN & PARR LLP/S.E.N.C.R.L., s.r.l.

By



Kendrick Lo  
Reg. No. 54,948  
Tel: 416-364-7311